Arduino Programming Cheat Sheet

Estrutura e Fluxo

Estrutura Básica do Programa

```
void setup() {
  // executado uma vez ao iniciar
}
void loop() {
  // executado repetidamente
}
```

Estruturas de Controle

```
if (x < 5) { ... } else { ... }
while (x < 5) { ... }
do { ... } while ( x < 5);
for (int i = 0; i < 10; i++) { ... }
break;  // sai do loop imediatamente
continue;  // para próxima interação
switch (myVar) {
   case 1:
     ...
     break;
   case 2:
     ...
     break;
   default:
     ...
}
return x; // return; para voids
```

Operadores

Operadores Básicos

```
=  (operador de atribuição)
+  (adição)      -  (subtração)
*  (multiplicação)  /  (divizão)
%  (modulo/resto)
== (igual a)    != (diferente de)
<  (menor que) >  (maior que)
<= (menor ou igual que)
>= (maior ou igual que)
&& (e)    || (ou)    ! (não)
```

Operadores Compostos

```
++ (incremento)
-- (decremento)
+= (adição composta)
-= (subtração composta)
*= (multiplicação composta)
/= (divisão composta)
&= (operador de bits e)
|= (operador de bits ou)
```

Operadores de Bits

```
& (e)                | (ou)
^ (nor)              ~ (não)
<< (mov. direita) >> (mov. esquerda)
```

Variáveis, Arrays e Dados

Tipo de Dado

```
void
boolean   (0, 1, true, false)
char      (e.g. 'a'  -128 to 127)
int       (-32768 até 32767)
long      (-2147483648 até 2147483647)
unsigned char  (0 até 255)
byte           (0 até 255)
unsigned int   (0 até 65535)
word           (0 até 65535)
unsigned long  (0 até 4294967295)
float     (-3.4028e+38 até 3.4028e+38)
double    (mesmo que float)
```

Qualificadores

```
static    (persiste entre as chamadas)
volatile  (na memória RAM)
const     (apenas leitura)
PROGMEM   (na memória flash)
```

Arrays

```
int myInts[6]; // array de 6 ints
int myPins[]={2, 4, 8, 3, 6};
int mySensVals[6]={2, 4, -8, 3, 2};
myInts[0]=42;  // atribui primeiro
               // índice de myInts
myInts[6]=12;  // ERRO! Índice
               // de 0 até 5
```

Constantes

```
HIGH  | LOW
INPUT | OUTPUT
true  | false
143        (Decimal)
0173       (Octal - base 8)
0b11011111 (Binário)
0x7B       (Hexadecimal - base 16)
7U         (Def sem sinal)
10L        (Def longo)
15UL       (Def longo sem sinal)
10.0       (Def flutuante)
2.4e5      (2.4*10^5 = 240000)
```

Acesso a Ponteiros

```
&  (reference: obter um ponteiro)
*  (dereference: seguir um ponteiro)
```

Strings

```
char S1[8] =
 {'A','r','d','u','i','n','o'};
 // sequência sem fim; pode falhar
char S2[8] =
 {'A','r','d','u','i','n','o','\0'};
 // inclui \0 terminação nula
char S3[]="Arduino";
char S4[8]="Arduino";
```

Funções Internas

I/O nos Pinos

```
Digital I/O (pinos: 0-13 A0-A5)
  pinMode(pin,[INPUT, OUTPUT])
  int digitalread(pin)
  digitalWrite(pin, valor)
    // escreve HIGH para a entrada
    // ativando resistores pull-up
Analog In (pinos: 0-5)
  int analogRead(pin)
  analogReference(
    [DEFAULT, INTERNAL, EXTERNAL])
PWM Out (pinos: 3 5 6 9 10 11)
  analogWrite(pin, valor)
```

I/O Avançado

```
tone(pin, freqhz)
tone(pin, freqhz, tempo_ms)
noTone(pin)
shiftOut(dataPin, clockPin,
  [MSBFIRST,LSBFIRST], value)
unsigned long pulseIn(pin,
  [HIGH,LOW])
```

Tempo

```
unsigned long millis()
  // overflow em 50 dias
unsigned long micros()
  // overflow em 70 minutos
delay(msec)
delayMicroseconds(usec)
```

Matemática

```
min(x, y)   max(x, y)   abs(x)
sin(rad)    cos(rad)    tan(rad)
sqrt(x)     pow(base, exponent)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
```

Números Aleatórios

```
randomSeed(seed) // long ou int
long random(max)
long random(min, max)
```

Bits e Bytes

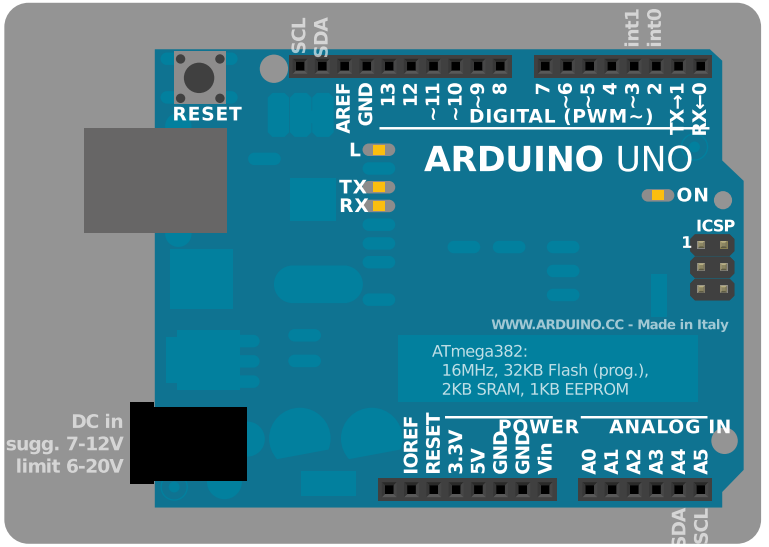
```
lowByte(x)    highByte(x)
bitRead(x, bitn)
bitWrite(x, bitn, bit)
bitSet(x, bitn)
bitClear(x, bitn)
bit(bitn)  // bitn: 0=LSB 7=MSB
```

Conversões de Tipo

```
char()        byte()
int()         word()
long()        float()
```

Interrupções Externas

```
attachInterrupt(interrupt, func,
 [LOW, CHANGE, RISING, FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()
```



Bibliotecas

Serial (comunicação com o PC ou via RX/TX)

```
begin(long Speed) // até 115200
end()
int available() //bytes disponíveis
byte read() //-1 nenhum disponível
byte peek()
flush()
print(myData)
println(myData)
write(myBytes)
SerialEvent() //
```

SoftwareSerial (serial comm. em qualquer pino)

```
  (#include <softwareSerial.h>)
SoftwareSerial(rxPin, txPin)
begin(long Speed) // até 115200
listen()       // Apenas 1
isListening() // por vez.
read, peek, print, println, write
```

EEPROM (#include <EEPROM.h>)

```
byte read(intAddr)
write(intAddr, myByte)
```

Servo (#include <Servo.h>)

```
attach(pin, [min_uS, max_uS])
write(angle)  // 0 to 180
writeMicroseconds(uS)
 // 1000-2000; 1500 é o ponto médio
int read()     // 0 até 180
bool attached()
detach()
```

Wire (I²C comm.) (#include <Wire.h>)

```
begin()      // join a master
begin(addr) // join a slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(myByte)            // Step 2
send(char * mystring)
send(byte * data, size)
endTransmission()       // Step 3
int available() //bytes disponíveis
byte receive()  // próximo byte
onReceive(handler)
onRequest(handler)
```